

```

/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     struct ListNode *next;
 * };
 */

struct ListNode *getIntersectionNode(struct ListNode *headA, struct ListNode *headB) {

    int cnt1 = 0;
    int cnt2 = 0;
    struct ListNode* l1 = headA;
    struct ListNode* l2 = headB;
    while(l1){
        cnt1++;
        l1 = l1->next;
    }
    while(l2){
        cnt2++;
        l2 = l2->next;
    }
    l1 = headA;
    l2 = headB;
    if(cnt1>cnt2){
        int dif = cnt1 - cnt2;
        while(dif){
            l1 = l1->next;
            dif--;
        }
    }
    while(l1 != l2){
        l1 = l1->next;
        l2 = l2->next;
    }
    return l1;
}

```

```

    }
}

if(cnt2>cnt1){
    int dif = cnt2 - cnt1;
    while(dif){
        l2 = l2->next;
        dif--;
    }
}

while((l1&& l2)&&(l1!=l2)){
    l1 = l1->next;
    l2 = l2->next;
}

return l1;
}

```

The screenshot displays a coding platform interface with the following components:

- Problem List:** Shows the problem name "Minimum Index Sum of Two Lists" with a difficulty level of "599".
- Submissions:** Indicates the submission is "Accepted" and was submitted by "Sohan_AK" on Jan 29, 2024 at 23:22.
- Performance Metrics:**
 - Runtime:** 33 ms, Beats 48.25% of users with C.
 - Memory:** 13.88 MB, Beats 50.98% of users with C.
- Code Editor:** Contains the C code for finding the intersection of two linked lists. The code uses two pointers, l1 and l2, and counts the number of nodes in each list (cnt1 and cnt2). It then moves the pointer with the higher count forward by the difference (dif) and traverses both lists until they intersect.
- Testcase:** Shows a single test case with the output "Intersected at 'B'" and the expected output "Intersected at 'B'".

Problem List

Run

Submit

Premium

Description

Editorial

Solutions

Submissions

All Submissions

Accepted

Soham_Ar submitted at Jan 29, 2024 23:22

Editorial

Solution

Runtime

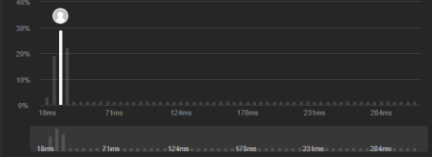
33 ms

Beats 48.25% of users with C

Memory

13.88 MB

Beats 93.90% of users with C



Graph showing runtime performance across different time intervals. The y-axis represents percentage (0% to 40%) and the x-axis represents time intervals (10ms to 204ms). A single data point is shown at 10ms, reaching approximately 33%.

Code: C

```
//  
* Definition for singly-linked list.  
* struct ListNode {  
*     int val;  
*     struct ListNode *next;  
* };
```

View more

More challenges

599. Minimum Index Sum of Two Lists

Write your action here

Code

Auto

Ln 47, Col 1

```
1 //  
2 * Definition for singly-linked list.  
3 * struct ListNode {  
4     int val;  
5     struct ListNode *next;  
6 };  
7  
8 struct ListNode *getIntersectionNode(struct ListNode *headA, struct ListNode *headB) {  
9  
10  
11     int cnt1 = 0;  
12     int cnt2 = 0;  
13     struct ListNode* l1 = headA;  
14     struct ListNode* l2 = headB;  
15     while(l1){  
16         cnt1++;  
17         l1 = l1->next;  
18     }  
19  
20     while(l2){  
21         cnt2++;  
22         l2 = l2->next;  
23     }  
24     l1 = headA;  
25     l2 = headB;  
26     if(cnt1>cnt2){  
27         int dif = cnt1 - cnt2;  
28         while(dif){  
29             l1 = l1->next;  
30             dif--;  
31         }  
32     }  
33 }
```

Saved to local

Testcase

Test Result

Input

8

[4,1,6,4,5]

[5,6,1,8,4,5]

2